MANAGER FOR INTEGRATING LANGUAGE TECHNOLOGY COMPONENTS

RELATED APPLICATIONS

[0001] This application claims priority under 35 U.S.C. § 119 based on U.S. Provisional Application No. 60/419,214 filed October 17, 2002, the disclosure of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

A. Field of the Invention

[0002] The present invention relates generally to automated speech recognition systems and, more particularly, to the creation of speech systems from basic speech and language technology components.

B. Description of Related Art

[0003] Many speech and language systems include a number of discrete language technology components that are in some way tied together by the overall system. Consider a speech indexing system that is designed to receive an audio signal, convert speech in the audio signal to a text transcription, and annotate the text to include additional information derived from the speech. This speech indexing system may include a number of individually designed speech and language technology components, such as a speech recognizer component, a speaker identification component, a topic detection component, and a name extraction component.

[0004] The speech recognizer component generates a basic text transcription of the speech. The other components operate on the transcription to generate additional

information that describes the transcription. Thus, the speaker identification component provides identifications of each speaker in the transcription, the topic detection component generates key words that define topics for segments of the transcription, and the name extraction component locates and mark all of the proper nouns in the transcription.

[0005] These basic language technology components may be reused in numerous other speech systems. Conventionally, when designing a new speech system, the designer manually integrates each of the basic language technology components into the system. This may involve individually controlling the start-up and handling of error conditions from the language technology components. Additionally, the designer may manually design structures for controlling the flow of data and control information between the language technology components. Significant resources may be spent on integrating the language technology components into the speech system.

[0006] Thus, there is a need in the art to improve the integration of basic language technology components into a high-level speech system.

SUMMARY OF THE INVENTION

[0007] Systems and methods consistent with the present invention include a Language Component Manager (LCM) that provides an interface between multiple language technology components and an external high-level application. Together, the high-level application, the LCM, and the language technology components create a complete speech system. The LCM marshals the interactions of the language technology components and presents a single system-level interface outside of the high-level

application. The LCM can help to significantly simplify and expedite the creation of speech systems that use the language technology components.

[0008] One aspect of the invention is directed to a method for interacting among components of a speech system that include language technology components, a middleware component, and at least one high-level application component. The method includes receiving substantially all data communications in the speech system at the middleware component and forwarding the data communications from the middleware component to a destination, one of the language technology components and the high-level application component, as determined by a configuration file. The method further includes receiving substantially all message communications in the speech system at the middleware component and forwarding the message communications from the middleware component to at least one of the language technology components and the high-level application component, as determined by the configuration file.

[0009] A second aspect consistent with the invention is directed to a speech system that includes a high-level application component, language technology components, a configuration file, and a language component manager (LCM). The configuration file describes communication paths among the high-level application component and the language technology components. The LCM acts as an intermediary for communications between the high-level application component and the language technology components and between the language technology components. The LCM connects the high-level application component and the language technology components based on the configuration file.

[0010] Yet another aspect consistent with the present invention is directed to a system integration device. The device includes ports configured to connect with first components that perform basic speech and language processing functions. The device also includes at least one port configured to connect with a high-level application designed to implement a speech service using the functions provided by the first components. Still further, the device includes a configuration file that includes routing information that describes communication paths among the high-level application and the first components. Communication between the first components and between the first components and the high-level application are routed through the device based on the routing information in the configuration file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate the invention and, together with the description, explain the invention. In the drawings,

[0012] Fig. 1 is a diagram illustrating an exemplary system in which concepts consistent with the invention may be implemented;

[0013] Fig. 2 is a diagram illustrating functional relationships of the components shown in Fig. 1;

[0014] Fig. 3 is a diagram that defines states of a Language Component Manager (LCM) consistent with an aspect of the invention;

[0015] Fig. 4 is a diagram illustrating communication paths through the components shown in Fig. 2;

[0016] Fig. 5 is a diagram illustrating a message transmitted between two components in a speech system;

[0017] Figs. 6A and 6B are diagrams illustrating an exemplary distributed application document; and

[0018] Fig. 7 is a diagram illustrating an exemplary speech application.

DETAILED DESCRIPTION

[0019] The following detailed description of the invention refers to the accompanying drawings. The same reference numbers may be used in different drawings to identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims and equivalents of the claim limitations.

[0020] A Language Component Manager (LCM) marshals the interactions (both message and data) of a group of speech and language technology components (LCs) and presents a single system-level interface to an outside application. The LCM takes on the responsibility of starting, shutting down, and restarting the LCs as necessary and it provides the high-level application with a well-defined system state. In addition, the LCM defines a framework for a fast system-abort function and it provides the LCs with a centralized logging facility.

SYSTEM OVERVIEW

[0021] Speech-related applications and systems, as described herein, may be performed on one or more processing devices or networks of processing devices. Fig. 1

is a diagram illustrating an exemplary system 100 in which concepts consistent with the invention may be implemented. System 100 includes a number of computing devices 101 that each include a computer-readable medium, such as random access memory 109, coupled to a processor 108. Computing devices 101 may also include a number of additional external or internal devices, such as, without limitation, a mouse, a CD-ROM, a keyboard, and/or a display.

[0022] Computing devices 101 may connect to a network 102. Network 102 may be, for example, a wide area network (WAN), such as the Internet, or a local area network (LAN). Certain ones of computing devices 101 may connect to network 102 and to each other through a router or switch 103.

[0023] In general, computing devices 101 may be any type of computing platform. Computing device 101 is exemplary only. Concepts consistent with the present invention can be implemented on any computing device, whether or not connected to a network.

[0024] Processors 108 can be any of a number of well-known computer processors, such as processors from Intel Corporation, of Santa Clara, California. Processors 108 execute program instructions stored in memory 109.

[0025] Memory 109 may contain application programs and data. In particular, as shown in Fig. 1, one of memories 109 may contain a high-level speech/language application 115. Application 115 may communicate with a Language Component Manager (LCM) 116 implemented in a manner consistent with the present invention. LCM 116 may include software that functions as middleware between high-level application 115 and basic speech and language technology components (LCs) 117. LCs 117 may include, for example, a speech recognizer component, a speaker identification

component, a topic detection component, a name extraction component, or other speech or language related components. Together, high-level application 115, LCM 116, and LCs 117 may form a complete speech system. Different high-level applications 115 may use different configurations of LCM 116 and LCs 117 to create different speech systems. High-level application 115 may, for example, use the services of LCs 117 to present an audio indexer speech system to end-users.

[0026] High-level application 115, LCM 116, and LCs 117 may inter-operate with one another in a distributed manner, as is illustrated in Fig. 1. In other implementations, application 115, LCM 116, and LCs 117 may be implemented on a single computing device.

[0027] Fig. 2 is a diagram illustrating functional relationships of a speech system 200 that is based on high-level application 115, LCM 116, and LCs 117 (labeled as LC1 through LCn). To high-level application 115, LCM 116 and LCs 117 appear as a single unified service. High-level application 115 does not directly communicate with LCs 117. Instead, high-level application 115 communicates with LCM 116, which acts as an intermediary between high-level application 115 and LCs 117. In this manner, LCM 116 presents a single system-level interface to application 115.

[0028] LCs 117 may include speech and language processing components directed to a wide variety of basic speech technologies. As previously mentioned, these speech technologies may include speech recognition, speaker identification, topic detection, and name extraction. Techniques for implementing these technologies are known in the art and will not be further described herein.

LCM STATES

[0029] LCM 116 provides high-level application 115 with a well-defined system state. The system state is a function of the system states of LCs 117. Fig. 3 is a diagram that defines the state machine for the LCM system state. LCM 116 may present one of five states to high-level application 115: down state 301, starting up state 302, aborting state 303, up state 304, and shutting down state 305.

[0030] On startup of LCM 116, the system state defaults to down state 301. In this state, LCs 117 are not running. High-level application 115 can interact with LCM 116 only through predefined administrative interfaces. The administrative interfaces may allow an administrator to log into LCM 116 via, for example, a web interface or a telnet interface. With the web interface, the administrator can interact with LCM 116 through a web browser. With the telnet interface, the administrator can interact with LCM 116 through a command-line based interface. In general, the administrative interfaces allow the administrator to perform functions, such as viewing error logging information, adjusting error logging verbosity, view a current configuration file, change the LCM configuration, and interact with the LCM by posting messages.

[0031] From down state 301, LCM 116 may receive a startup request through the administrative interfaces. In response to the startup request, LCM 116 starts LCs 117 and enters starting up state 302. LCs 117 may be started based on initialization information contained in a configuration file, called a distributed application document. When started, LCs 117 may boot-up and open a communication channel with LCM 116. If any of LCs 117 fail to connect within a pre-specified timeout period, LCM 116 assumes a

non-recoverable failure of the LC 117 and changes the system state to shutting down state 305.

[0032] When an LC 117 reaches a state of operational readiness (e.g., it has loaded and processed its model data), it reports its state to LCM 116. LCM 116 may respond by activating data and message connections. When all of LCs 117 are ready, LCM 116 enters up state 304.

[0033] Up state 304 is the primary operational state of LCM 116. In up state 304, LCM 116 is responsible for managing data and message communications between LCs 117 and high-level application 115. While in up state 304, LCM 116 may monitor LCs 117 for failure and switch its state to shutting down state 305 in the event of a failure.

[0034] In shutting down state 305, LCM 116 may discontinue data and message routing activities between the various components (i.e., LCs and high-level applications). LCM 116 may then disconnect any connected high-level applications 115 and subsequently instruct all LCs 117 to shut down. When these actions are complete, LCM 116 may enter down state 301.

[0035] Administrators or high-level applications 115 may issue abort requests to LCM 116 to disconnect from LCM 116. LCM 116 may respond by entering aborting state 303. In this state, LCM 116 deactivates message and data routing between the LCs 117 and the aborting high-level application or administrator.

[0036] By providing a centralized system state, LCM 116 can control instabilities due to random timing problems between LCs 117. More particularly, because LCM 116 controls all communications between components in system 200, LCM 116, in the event

of an error, can shut down the components in an orderly manner and, thus, avoid system race conditions.

COMPONENT COMMUNICATION

[0037] As previously mentioned, LCs 117 and high-level application 115 do not communicate directly with one another. Instead, all communications in speech system 200 are passed through LCM 116, which defines a framework for inter-component communication. Fig. 4 is a diagram illustrating communication paths in system 200.

[0038] As shown in Fig. 4, LC1 communicates with LCn through LCM 116. Similarly, high-level application 115 and LC1-LCn communicate with one another through LCM 116. A benefit of this communication framework is that it minimizes port management inside the LCs and the high-level applications, such as application 115. Specifically, LCs 117 and high-level applications 115 only need to communicate with one entity — LCM 116.

[0039] The LCM communication framework may provide two ways for communicating: message and data. Message communication can be conceptualized as an asynchronous remote procedure call by an initiating component to a receiving component. Message communication is generally used to initiate actions in LCs 117. Data communication provides a way to pass data units between components, where these data units do not invoke actions in the receiving component. Instead, the data units are processed by the receiving component.

[0040] Generally, the purpose of sending messages is to invoke an action in the receiver of the message. In one implementation, messages are transmitted as serialized

extensible markup language (XML) elements. The end of a message may be indicated by a zero byte (i.e. 0x00, a.k.a. NULL-termination). Fig. 5 is a diagram illustrating a message 501 transmitted between two LCs 117, labeled as LC1 and LC2. Message 501 is a simple message that instructs LC2 to take a particular action.

[0041] Message 501 has a one-to-one relationship between components, i.e., there is one sender (LC1) and one receiver (LC2) of the message. LCM 116 may also support *m*:*n* message relationships, in which there can be *m* senders and *n* receivers of one particular message, where both *m* and *n* are greater than or equal to one. A message can be sent from any one of the *m* senders for delivery to each of the *n* receivers.

[0042] Table I, below, illustrates a number of possible system messages that relate to the control of LCM 116.

TABLE I

Message Name	Description
LCM_ABORT	While executing a system-abort operation, the LCM will send an <LCM_ABORT/> to every component (LCs and high-level applications), instructing them to discontinue any system-related activities, reset themselves and then report their readiness.
LCM_ABORT_DONE	Once a component has reacted appropriately to an abort request from the LCM, it notifies the LCM of its return to operational readiness by sending an <LCM_ABORT_DONE/>.
LCM_CONNECT	After a high-level application opens a socket connection to the LCM's port, it sends an <LCM_CONNECT name=""/>, identifying itself using the attribute name.
LCM_CONNECT_ACK	In response to an LCM_CONNECT message from a high-level application, the LCM will send an LCM_CONNECT_ACK to the high-level application.
LCM_DISCONNECT	The LCM sends an <LCM_DISCONNECT/> message to a high-level application to instruct the high-level application to disconnect.
LCM_HELLO	After an LC opens a socket connection to the LCM's port, it will send an <LCM_HELLO process-id=""/>, identifying itself via its process ID in the attribute process-id.
LCM_HELLO_ACK	In response to an LCM_HELLO message from an LC, the LCM will send back an LCM_HELLO_ACK.
LCM_LOG	This message is sent from LCs to the LCM providing text that the respective LC wishes to add to the system log. The message syntax is: <LCM_LOG level="..." message="..."/>, where the attribute level specifies the logging level, and message contains the logging text.

LCM_READY	An LC sends an <LCM_READY/> message to the LCM when it has reached operational readiness.
LCM_SHUTDOWN	The LCM will send an <LCM_SHUTDOWN/> message to an LC to instruct the LC to shut itself down.
LCM_SYSTEM_ABORT	A high-level application can send an <LCM_SYSTEM_ABORT/> message to the LCM to initiate a system-abort.
LCM_SYSTEM_READY	An <LCM_SYSTEM_READY/> will be distributed to all LCs and (potentially connected) high-level applications, when the entire system has reached a state of operational readiness, i.e. when the system state is changed to *up*.

[0043] Generally, the purpose of data communication between components, such as LCs 117 and high-level applications 115, is to pass data units, such as packets, of structured information between the components. In contrast to messages, data communication does not generally invoke actions in the receiving component. Instead, the receiving component consumes or processes the information contained in the packets.

[0044] In one implementation consistent with the invention, a data pipe model is used to pass data between components. The data pipe may be a one-to-one relationship between the component that sends/produces data and the component that receives/consumes the data. A data pipe may be implemented by the producer by sending the following pieces of information to LCM 116: the name of the data pipe, a data pipe command, and any arguments required by the data pipe command. LCM 116 will then deliver the data pipe command and its arguments (if any) to the appropriate data-pipe consumer. A data pipe may be configured to connect multiple LCs 117 in series. In this manner, data produced by a first LC may be consumed by a second LC, which may then produce a modified version of the data which may then be consumed by a third LC.

[0045] Table II, below, lists three exemplary data pipe commands.

TABLE II				
Command	**Command ID**	**Arguments**	**Effect**	**Return-value (besides error code)**
Clear data-pipe	3	None	Instruct the data consumer to remove any unprocessed data packets that it has in its data-receiving buffer	None
Get number of elements in data-pipe buffer	4	None	Instruct the data consumer to return the number of data packets that are currently held in the data-receiving buffer.	Unsigned integer, indicating the number of data packets that are currently in the data-receiving buffer.
Send data	1	Data packet	Delivers a data packet to the data-pipe consumer. The data-packet can be rejected by the data-consumer using the error code *BufferFull*. In this case, the data producer is responsible for retransmitting the data packet at a later time.	None

DISTRIBUTED APPLICATION DOCUMENT

[0046] For a particular high-level application 115, the choice of which LCs 117 to use in the application and the paths for the flow of messages and data between LCs 117 and the high-level application 115 is controlled by a configuration document (or file) referred to herein as a distributed application document (DAD).

[0047] In one implementation, the DAD is an XML document that describes in detail the configuration of an LCM-managed system. LCM 116 parses the DAD in determining the configuration of speech system 200. A particular system configuration may define the specification of LCs 117, the specification of high-level application 115, the message

and data communication connections between all the components, and error logging settings. As such, the DAD can be viewed as a complete system specification.

[0048] Figs. 6A and 6B illustrate an exemplary DAD 600. For ease of explanation, the DAD shown in Fig. 6 is a simplified version of a typical DAD.

[0049] DAD 600 includes a name tag 601 that includes the name of the particular high-level application. In this example, the application is "Audio Indexer."

[0050] Run-time section 602 may include, during run-time, a list of all connected LCs 117 and high-level applications 115. These run-time connections may be dynamically changed by LCM 116 as the connected components change. In this example, run-time section 602 also includes an auto-start tag that indicates that DAD 600 is to be automatically loaded by LCM 116 when LCM 116 is initially started.

[0051] The system specification section, located between the tags "<SYSTEM_SPECIFICATION>" (Fig. 6A) and "</SYSTEM_SPECIFICATION>" (Fig. 6B), includes a complete description of the configuration of the components in speech system 200, including the message and data paths. As shown, the "SYSTEM_SPECIFICATION" section includes a "CONNECTION_LISTENERS" section, a "LOGGING_TARGETS" section, a "MESSAGES" section, a "DATA_PIPES" section (Fig. 6B), a "REMOTE_APPLICATIONS" section (Fig. 6B), and a "LANGUAGE_COMPONENTS" section (Fig. 6B). Each of these sections is illustrated in Fig. 6 by corresponding XML tags in which brackets ("< . . . >") denote the beginning of a section and brackets with a slash ("</ . . . >") denote the end of a section.

[0052] The CONNECTION_LISTENERS section includes a number of CONNECTION_LISTENER tags that define the low-level connections between the

components, including the port of LCM 116 with which the components connect. Additionally, these tags may also define whether each port is a port dedicated to a data producer, data consumer, or message.

[0053] The LOGGING_TARGETS section defines error logging settings for the system, such as target file names. Additionally, the level of detail to which errors are logged can be set in this section through the verbosity setting. Valid values for the verbosity setting according to one implementation consistent with the invention are shown in Table III.

TABLE III

Logging Level	Description
None	No logging information.
Error	Indicates a severe exception condition in the LC. This condition often leads to component failure and hence to system shutdown.
Warning	Unexpected exception-like condition. The LC can avoid failure but this is still a severe exception.
Info	Provides high-level component state information that might be of general interest.
Detail	Provide very fine-grain detailed information about the component's system state. This type of information is typically useful for component debugging.

[0054] The MESSAGES section defines the possible flow of messages between the components in speech system 200. The MESSAGES section may include three sub-sections: "PUBLIC_IN", "PUBLIC_OUT", and "PRIVATE." PUBLIC_IN refers to messages generated by high-level application 115 and transmitted to LCM 116. PUBLIC_OUT refers to messages from LCM 116 and transmitted to high-level applications 115. PRIVATE refers to messages transmitted between LCs 117 via LCM 116.

[0055] Exemplary message definitions are illustrated for the PUBLIC_OUT sub-section of MESSAGES. Messages in PUBLIC_IN and PRIVATE may be implemented

similarly. Each entry 604 in the PUBLIC_OUT sub-section includes a message name ("message name" field), the name of the LC from which the message originates ("from" field), and the name of the destination high-level application ("to") field. In this example, the messages are all destined for the application "decode." The messages named "passage" and "gap" originate from the LCs identified as "spk_segment." The message named "word" originates from the LC named "nbscore." In operation, the LC named "nbscore" may transmit a message to the LCM that includes the label "word." LCM 116, based on DAD 600, knows that this message should be forwarded to the high-level application "decode."

[0056] The DATA_PIPES section defines the possible flow of data between the components in system 200. As with the MESSAGES section, the DATA_PIPES section may include the three sub-sections PUBLIC_IN, PUBLIC_OUT, and PRIVATE. These sections similarly define data pipes between high-level applications 115 and LCM 116, between LCM 116 and the high-level applications, and between LCs, respectively.

[0057] Two exemplary data pipe entries are shown in DAD 600 under the PRIVATE sub-section of DATA_PIPES. The first data pipe entry is named "phn2seq." This data pipe is received by LCM 116 from the LC "filter_component" and forwarded to the LC "speaker_seg." The second data pipe entry is named "seg2fw." Data transmitted through this data pipe is received by LCM 116 from the LC "speaker_seg" and forwarded to the LC "decoder_fw."

[0058] The "REMOTE_APPLICATIONS" section defines high-level applications 115 that connect to LCM 116. One remote application, named "feed_audio" is illustrated as being defined in this section.

[0059] The "LANGUAGE_COMPONENTS" section defines LCs 117 that may communicate with LCM 116. One LC 117, named "decoder_phn" is defined in this section.

EXEMPLARY APPLICATION

[0060] Fig. 7 is a diagram illustrating components in an exemplary speech application. The speech application shown in Fig. 7 may be a dialog manager that recognizes speech received over a telephone connection and takes action based on the instructions from the telephone caller.

[0061] High-level application 701 may include a dialog manager application that responds to speech-based commands from the caller and generates appropriate responses for the caller. The speech processing used to recognize the commands in the speech is primarily performed by LCs 703-707. LCs 703-707 include DTMF (dual-tone multi-frequency) recognizer component 703, voice-over-IP (VoIP) component 704, speech segmenter component 705, text-to-speech component 706, and speech recognizer component 707.

[0062] DTMF component 703 is configured to recognize key presses on a touch-tone telephone based on the corresponding DTMF signal generated by the telephone. VoIP component 704 receives the user's telephone call. In this example, the telephone call is received as a packetized signal in a VoIP connection. Speech segmenter component 705 segments speech signals into useful segments. Speech segmenter component 705 may, for example, segment input speech signals into audio segments corresponding to individual words or sentences. Text-to-speech component 706 may generate synthesized

speech signals based on input text documents. Speech recognizer component 707 performs basic speech recognition of input speech signals.

[0063] LCM 702 acts as an intermediary between high-level application 701 and LCs 703-707. LCM 702 reads a DAD document to determine the message and data interactions between high-level application 701 and LCs 703-707. For example, the DAD may dictate that speech received at VoIP component 704 is to be transmitted via a data pipe to speech segmenter component 705. The output of speech segmenter component 705 may be forwarded to speech recognizer component 707, the output of which may then be forwarded to high-level application 701. Similarly, high-level application 701 may respond to callers by generating text that it transmits, via LCM 702, to text-to-speech component 706. The synthesized text output from text-to-speech component 706 may then be transmitted to VoIP component 704 for transmission to the caller. In each of these instances of data transfer, the data is transmitted through and routed by LCM 702.

CONCLUSION

[0064] An LCM, as described above, functions as middleware between one or more language and technology components and one or more high-level applications. In aggregate, the LCM, the language and technology component(s), and the high-level application(s) form a speech system. The components of the speech system may be distributed. All data and messages in the speech system pass through the LCM. A configuration file defines relationships, such as message and data paths, for the speech system.

[0065] The foregoing description of preferred embodiments of the invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention.

[0066] Certain portions of the invention have been described as software that performs one or more functions. The software may more generally be implemented as any type of logic. This logic may include hardware, such as application specific integrated circuit or a field programmable gate array, software, or a combination of hardware and software.

[0067] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used.

[0068] The scope of the invention is defined by the claims and their equivalents.